Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

MOTOROLA

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Embedded SDK (Software Development Kit)

V.22bis Library

SDK108/D
Rev. 2, 07/23/2002

© Motorola, Inc., 2002. All rights reserved.

MOTOROLA intelligence everywhere™
digital dna™

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Contents

About This Document

Audience ix
Organization ix
Suggested Reading ix
Conventions x
Definitions, Acronyms, and Abbreviations x
References xi

Chapter 1
Introduction

1.1 Quick Start 1-1
1.2 Overview of V.22 bis 1-1
1.2.1 Background 1-2
1.2.2 Features and Performance 1-2

Chapter 2
Directory Structure

2.1 Required Core Directories 2-1
2.2 Optional (Domain-Specific) Directories 2-2

Chapter 3
V.22bis Library Interfaces

3.1 V.22bis Services 3-1
3.2 Interface 3-1
3.3 Specifications 3-4
3.3.1 v22bisCreate 3-5
3.3.2 v22bisInit 3-6
3.3.3 v22bisTXDataInit 3-11
3.3.4 v22bisTX 3-13
3.3.5 v22bisRX 3-15
3.3.6 v22bisControl 3-17
3.3.7 v22bisDestroy 3-18

Chapter 4
Building the V.22bis Library

4.1 Building the V.22bis Library 4-1
4.1.1 Dependency Build 4-1
4.1.2 Direct Build 4-2

For More Information On This Product,
Go to: www.freescale.com

Chapter 5
Linking Applications with the V.22bis Library

5.1 V.22bis Library 5-1
5.1.1 Library Sections 5-1

Chapter 6
V.22bis Applications

6.1 V.22bis Test and Demo Applications 6-1

Chapter 7
License

7.1 Limited Use License Agreement 7-1

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

List of Tables

Table 1-1 V.22bis Operating Range 1-2
Table 3-1 v22bisCreate Arguments. 3-5
Table 3-2 v22bisInit Arguments 3-6
Table 3-3 v22bisTXDataInit Arguments. 3-11
Table 3-4 v22bisTX Arguments 3-13
Table 3-5 v22bisRX Arguments 3-15
Table 3-6 v22bisControl Arguments. 3-17
Table 3-7 v22bisDestroy Arguments 3-18

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

List of Figures

Figure 2-1 Core Directories 2-1
Figure 2-2 modem Directory 2-2
Figure 2-3 V22bis Directory Structure 2-2
Figure 2-4 V22bis applications Directory 2-3
Figure 4-1 Dependency Build for V.22bis Library 4-1
Figure 4-2 v22bis.mcp Project 4-2
Figure 4-3 Execute Make 4-2

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

List of Examples

Code Example 3-1 C Header File v22bis.h 3-1
Code Example 3-2 Sample Transmit Callback Procedure. 3-8
Code Example 3-3 Sample Receive Callback Procedure 3-8
Code Example 3-4 Use of the v22bisInit Interface 3-9
Code Example 3-5 Use of v22bisTXDataInit Interface. 3-11
Code Example 3-6 Use of v22bisTX Interface 3-14
Code Example 3-7 Use of v22bisRX Interface 3-15
Code Example 3-8 Use of v22bisDestroy Interface 3-18
Code Example 5-1 V.22bis Library Sections 5-2

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

About This Document

This manual describes the V.22bis telecommunication algorithm for use with Motorola's Embedded Software Development Kit (SDK).

Audience

This document targets software developers implementing communications functions within software applications.

Organization

This manual is arranged in the following sections:

- **Chapter 1, Introduction**—provides a brief overview of this document
- **Chapter 2, Directory Structure**—provides a description of the required core directories
- **Chapter 3, V.22bis Library Interfaces**—describes all of the V.22bis Library functions
- **Chapter 4, Building the V.22bis Library**—tells how to execute the system library project build
- **Chapter 5, Linking Applications with the V.22bis Library**—describes organization of the V.22bis Library
- **Chapter 6, V.22bis Applications**—describes the use of V.22bis Library through test/demo applications
- **Chapter 7, License**—provides the license required to use this product

Suggested Reading

We recommend that you have a copy of the following references:

- *DSP56800 Family Manual*, DSP56800FM/AD
- *DSP568xx User's Manual*, for the DSP device being implemented
- *Inside CodeWarrior: Core Tools*, Metrowerks Corp.

Freescale Semiconductor, Inc.
For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Conventions

This document uses the following notational conventions:

Typeface, Symbol or Term	Meaning	Examples
`Courier Monospaced Type`	Commands, command parameters, code examples, expressions, datatypes, and directives	...`*Foundational include files`... ...a data structure of type `vad_tConfigure`...
Italic	Calls, functions, statements, procedures, routines, arguments, file names and applications	...the *pConfig* argument... ...defined in the C header file, *aec.h*... ...makes a call to the *Callback* procedure...
Bold	Reference sources, paths, emphasis	...refer to the **Targeting DSP56824 Platform** manual.... ... see: **C:\Program Files\Motorola\Embedded SDK\help\tutorials**
Bold/Italic	Directory name, project name	...and contains these core directories: ***applications*** contains applications software.... ...CodeWarrior project, ***3des.mcp***, is.....
Blue Text	Linkable on-line	...refer to **Chapter 7**, License...
Number	Any number is considered a positive value, unless preceded by a minus symbol to signify a negative value	3V -10 DES^{-1}
ALL CAPITAL LETTERS	Variables, directives, defined constants, files libraries	INCLUDE_DSPFUNC #define INCLUDE_STACK_CHECK
Brackets [...]	Function keys	...by pressing function key [F7]...
Quotation marks "... "	Returned messages	...the message, "Test Passed" is displayed.... ...if unsuccessful for any reason, it will return "NULL"....

Definitions, Acronyms, and Abbreviations

The following list defines the acronyms and abbreviations used in this document. As this template develops, this list will be generated from the document. As we develop more group resources, these acronyms will be easily defined from a common acronym dictionary. Please note that while the acronyms are in solid caps, terms in the definition should be initial capped ONLY IF they are trademarked names or proper nouns.

DSP Digital Signal Processor or Digital Signal Processing

I/O Input/Output

IDE Integrated Development Environment

LSB Least Significant Bit

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

MAC	Multiply/Accumulate
MIPS	Million Instructions Per Second
MSB	Most Significant Bit
OnCE™	On-Chip Emulation
OMR	Operating Mode Register
PC	Program Counter
SDK	Software Development Kit
SP	Stack Pointer
SPI	Serial Peripheral Interface
SR	Status Register
SRC	Source

References

The following sources were used to produce this book:

1. *DSP56800 Family Manual, DSP56800FM/AD*
2. *DSP568xxUser's Manual*
3. *Embedded SDK Programmer's Guide*

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Chapter 1
Introduction

Welcome to Motorola's Family of Digital Signal Processors (DSPs). This document describes the V.22bis Library, which is a part of Motorola's comprehensive Software Development Kit (SDK) for its DSPs. In this manual, you will find all the information required to use and maintain the V.22bis Library interface and algorithms.

Motorola provides these algorithms to you for use on the Motorola Digital Signal Processors to expedite your application development and reduce the time it takes to bring your own products to market.

Motorola's V.22bis Library is licensed for your use on Motorola processors. Please refer to the standard Software License Agreement in **Chapter 7** for license terms and conditions; please consult with your Motorola representative for premium product licensing.

1.1 Quick Start

Motorola Embedded SDK is targeted to a large variety of hardware platforms. To take full advantage of a particular hardware platform, use **Quick Start** from the **Targeting Motorola DSP568xx Platform** documentation.

For example, the **Targeting Motorola DSP56824 Platform** manual provides more specific information and examples about this hardware architecture. If you are developing an application for a DSP56824EVM board or any other DSP56824 development system, refer to the **Targeting Motorola DSP56824 Platform** manual for **Quick Start** or other DSP56824-specific information.

1.2 Overview of V.22 bis

V.22bis conforms to the V.22 bis Low-Speed Modem ITU-T Standard.

Lower-end internet telephony and e-cash applications require a low-speed modem for exchanging data. The ITU-T standard V.22bis with 2400 bits per second is usually the preferred modem for these applications. Motorola has developed internally a single library module containing the V.22 bis modem. Using a network simulator, the modem was tested over all possible telephone networks available in continental North America and is available for the Motorola DSP568xx range of processors.

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

1.2.1 Background

The ITU-T V.22bis recommendation describes a split-band modem for use on the General Switched Telephone Network and on Point-to-Point two wire leased telephone-type circuits. A split-band modem is characterized by transmission and reception on two spectral bands which do not overlap. The standard defines:

- Calling modem to:
  - transmit with a carrier frequency of 1200Hz and
  - receive with a carrier frequency of 2400Hz
- Answering modem to:
  - transmit with a carrier frequency of 2400Hz and
  - receive with a carrier frequency of 1200Hz

Quadrature amplitude modulation technique is used for each channel with synchronous line transmission at 600 baud. The constellation could be either 16 point, 4 bits/baud supporting an input bit rate of 2400bps, or 4 point, 2 bits/baud supporting 1200bps. A scrambler is included in the input to the transmitter and a descrambler at the output of the receiver. The modem should have both an adaptive equalizer and a compromise equalizer. A guard tone of 1800 ±20Hz or 550 ±20Hz may be used while transmitting only in the high channel (transmitter of the answering modem).

1.2.2 Features and Performance

Table 1-1 shows the operating range of the V.22bis modem.

Table 1-1. V.22bis Operating Range

Data Rate	2400 / 1200 bits per second
Line Bandwidth	300 - 3400Hz
Sampling Rate	7200Hz

The V.22bis library is not multichannel.

For details on Memory and MIPS for a particular DSP, refer to the **Libraries** chapter of the appropriate Targeting manual.

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Chapter 2
Directory Structure

2.1 Required Core Directories

Figure 2-1 details required platform directories:

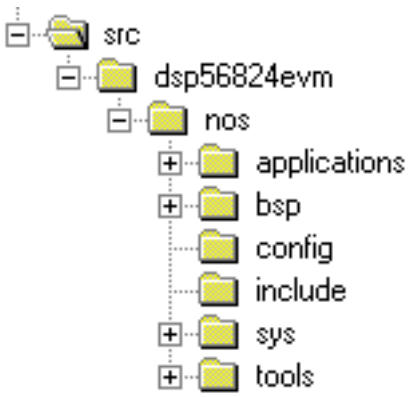

Figure 2-1. Core Directories

As shown in **Figure 2-1**, DSP56824EVM has no operating system (nos) support and contains the following core directories:

- ***applications*** contains applications software that can be exercised on this platform
- ***bsp*** contains board support package specific for this platform
- ***config*** contains default HW/SW configurations for this platform
- ***include*** contains SDK header files which define the Applicatin Programming Interface
- ***sys*** contains required system components
- ***tools*** contains utilities used by system components

There are also optional directories that include domain-specific libraries.

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

2.2 Optional (Domain-Specific) Directories

Figure 2-2 demonstrates how the V.22bis algorithm is encapsulated in the domain-specific directory *modem*.

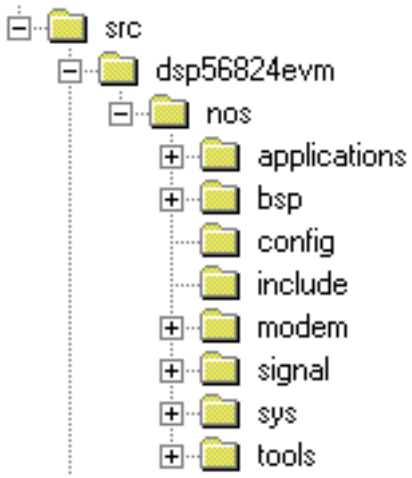

Figure 2-2. *modem* Directory

The *modem* directory includes such modem-specific algorithms as V.22bis, V.42bis, V.8bis and others. The *v22* directory includes the V.22bis-specific algorithms. Figure 2-3 shows the directories specific to V.22bis, found in the *v22* directory.

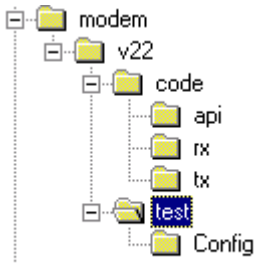

Figure 2-3. V22bis Directory Structure

The ***code*** sub-directory under the *v22* directory includes the following:

- ***api*** includes both C and assembly API files for the V.22bis library
- ***rx*** contains Assembly source files to implement the V.22bis receiver
- ***tx*** contains Assembly source files to implement the V.22bis transmitter

The ***test*** sub-directory under the *v22* directory includes the following:

- ***Config*** includes the configuration files *appconfig.h*, *appconfig.c*, and *linker.cmd* for testing the V.22bis library.

For More Information On This Product,
Go to: www.freescale.com

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

The *applications* directory includes high-level software that exercises these modem libraries. For example, the application *v22_demo* shown in **Figure 2-4**, can interface the V.22bis library with the PC application hyperterminal to produce user-specified functionality.

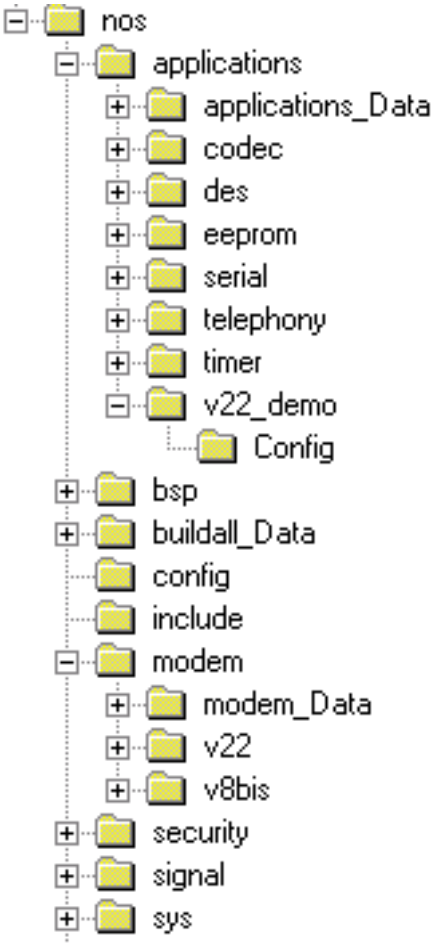

Figure 2-4. V22bis *applications* Directory

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Chapter 3
V.22bis Library Interfaces

3.1 V.22bis Services

V.22bis provides the data pump service which modulates the bits to be transmitted and demodulates the bits to be received. This library accepts as input a stream of bits at 1200/2400bps, modulates and calls back the application, providing 12 samples/call back at a sampling rate of 7200 samples/second. The user application must transmit these samples to the codec at 7200 samples/second. Similarly, the receiver accepts the samples at 7200 samples/second, demodulates and calls back the user application, providing a byte (8 bits) per call back. The V.22bis library can be accessed through these APIs: *create*, *transmit*, *receive* and *destroy*.

3.2 Interface

The C interface for V.22bis library services is defined in the C header file *v22bis.h*, shown in **Code Example 3-1**.

Code Example 3-1. C Header File *v22bis.h*

```
/* File: v22bis.h */

#ifndef __V22bis_H
#define __V22bis_H

/**************************
 Foundational Include Files
**************************/
#include "port.h"

/****************************************
 #define for V22bis configuration flags
****************************************/

#define  V22BIS_CALL_MODEM              0x0000
#define  V22BIS_ANSWER_MODEM            0x0001
```


For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

```
#define  V22BIS_GUARD_TONE_ENABLE      0x0002
#define  V22BIS_GUARD_TONE_DISABLE     0x0000
#define  V22BIS_GUARD_TONE_550Hz       0x0000
#define  V22BIS_GUARD_TONE_1800Hz      0x0004
#define  V22BIS_SELF_RETRAIN_ENABLE    0x0008
#define  V22BIS_SELF_RETRAIN_DISABLE   0x0000
#define  V22BIS_ENABLE_2100_ANSWERTONE  0x0000
#define  V22BIS_DISABLE_2100_ANSWERTONE 0x2000
#define  V22BIS_V14_ENABLE_ASYNC_MODE  0x4000
#define  V22BIS_V14_DISABLE_ASYNC_MODE 0x0000
#define  V22BIS_LOOPBACK_ENABLE        0x8000
#define  V22BIS_LOOPBACK_DISABLE       0x0000


/*****************************************
     Structure for V22bis Status
*****************************************/


typedef enum
{
      V22BIS_1200BPS_CONNECTION_ESTABLISHED,
      V22BIS_2400BPS_CONNECTION_ESTABLISHED,
      V22BIS_CONNECTION_LOST,
      V22BIS_DATA_AVAILABLE,
      V22BIS_RETRAINING,
      V22BIS_CARRIER_LOST_IN_DATAMODE
} v22bis_eStatus;


typedef struct
{
             void    (*pCallback) (  void * pCallbackArg,
                          v22bis_eStatus   Status,
                          char * pBits,
                          UWord16 NumberBits);
                          void * pCallbackArg;
} v22bis_sRXCallback;


typedef struct
{
             void    (*pCallback) (  void * pCallbackArg,
                          v22bis_eStatus Status,
                          Word16 * pSamples,
                          UWord16 NumberSamples);
                          void * pCallbackArg;
} v22bis_sTXCallback;



/* Please refer to the #defines above which go into
   Flags mentioned in the structure define below */
typedef struct
{
```


For More Information On This Product,
Go to: www.freescale.com

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

```
            UWord16             Flags;
            v22bis_sTXCallback  TXCallback;
            v22bis_sRXCallback  RXCallback;
} v22bis_sConfigure;



/* This structure is not currently used; it
 * is given here for future use. */
typedef struct
{
            UWord16 Flags;
} v22bis_sHandle;

/**************************
 Function Prototypes
**************************/

EXPORT v22bis_sHandle * v22bisCreate ( v22bis_sConfigure * pConfig);
EXPORT Result           v22bisInit   ( v22bis_sHandle * pV22bis,
                                       v22bis_sConfigure * pConfig);
EXPORT Result           v22bisTXDataInit ( v22bis_sHandle * pV22bis,
                                           char  * pBits,
                                           UWord16  NumBytes);
EXPORT Result           v22bisTX     (v22bis_sHandle * pV22bis);

EXPORT Result           v22bisRX     (v22bis_sHandle * pV22bis,
                                         Word16  * pSamples,
                                         UWord16 NumberSamples);
EXPORT Result           v22bisDestroy( v22bis_sHandle * pV22bis);
EXPORT Result           v22bisControl (UWord16  Command);

#endif
```

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

3.3 Specifications

The following pages describe the V.22bis library functions.

Function arguments for each routine are described as *in*, *out*, or *inout*. An *in* argument means that the parameter value is an input only to the function. An *out* argument means that the parameter value is an output only from the function. An *inout* argument means that a parameter value is an input to the function, but the same parameter is also an output from the function.

Typically, *inout* parameters are input pointer variables in which the caller passes the address of a preallocated data structure to a function. The function stores its results within that data structure. The actual value of the *inout* pointer parameter is not changed.

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

For More Information On This Product,
Go to: www.freescale.com

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

3.3.1 *v22bisCreate*

Call(s):

```
v22bis_sHandle *v22bisCreate (v22bis_sConfigure *pConfig);
```

Required Header: *v22bis.h*

Arguments:

Table 3-1. *v22bisCreate* Arguments

pConfig	in	Points to the configuration data for V.22bis

Description: The *v22bisCreate* function creates an instance of the V.22bis data pump. During the *v22bisCreate* call, any dynamic resources required by the V.22bis algorithm are allocated. The current V.22bis implementation is NOT re-entrant, so multiple instances are NOT supported. One word is allocated per instance.

The *pConfig* argument points to the *v22bis_sConfigure* structure used to configure the V.22bis operation. To initialize V.22bis operation, *v22bisCreate* calls *v22bisInit*. Please see *v22bisInit*, **Section 3.3.2**, for a description of the *v22bis_sConfigure* fields.

Returns: Upon successful completion, the *v22bisCreate* function will return a pointer to the specific instance of V.22bis created. If *v22bisCreate* is unsuccessful for any reason, it will return “NULL”.

Special Considerations: The current implementation of V.22bis requires memory sections to be allocated via linker commands. For details on linking applications, see **Chapter 5**.

Code Example: See **Code Example 3-5.**

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

3.3.2 *v22bisInit*

Call(s):

```
Result v22bisInit (v22bis_sHandle *pV22bis, v22bis_sConfigure *pConfig);
```

Required Header: *v22bis.h*

Arguments:

Table 3-2. *v22bisInit* Arguments

pV22bis	in	Pointer to an instance of V.22 bis
pConfig	in	A pointer to a data structure containing data for initializing the V.22bis algorithm

Description: The *v22bisInit* function will initialize the V.22bis data pump algorithm. During initialization, all resources will be set to their initial values in preparation for V.22bis operation.The current V.22bis implementation is NOT re-entrant, so multiple instances are NOT supported. The library itself allocates static memory for V.22bis operation, so the user need not allocate memory for the library.

The parameter *pConfig* points to a data structure of the type *v22bis_sConfigure*; its fields initialize V.22bis operation in the following manner:

Flags A set of configuration options for the V.22bis data pump. These flags are *OR'ed* together to produce the Flags' parameter value. Flags options include:

V22BIS_CALL_MODEM - Makes V.22bis a calling modem

V22BIS_ANSWER_MODEM - Makes V.22bis an answering modem

V22BIS_GUARD_TONE_ENABLE - Enables the guard tone

V22BIS_GUARD_TONE_DISABLE - Disables the guard tone

V22BIS_GUARD_TONE_550Hz - Sets the guard tone at 550Hz

V22BIS_GUARD_TONE_1800Hz - Sets the guard tone at 1800Hz

V22BIS_SELF_RETRAIN_ENABLE - Reestablishes a connection if lost

V22BIS_SELF_RETRAIN_DISABLE - Does not reestablish a lost connection

V22BIS_ENABLE_2100_ANSWERTONE - Sends/Expects 2100Hz answer tone

V22BIS_DISABLE_2100_ANSWERTONE 0x2000 - Does not send/expect 2100Hz answer tone

V22BIS_V14_ENABLE_ASYNC_MODE - Enables padding start & stop bits

V22BIS_V14_DISABLE_ASYNC_MODE - No start & stop bits, plain data

V22BIS_LOOPBACK_ENABLE - Enables loopback mode for testing

V22BIS_LOOPBACK_DISABLE - Disables test loopback mode

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

For More Information On This Product,
Go to: www.freescale.com

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

pTXCallback A pointer to a *v22bis_sTXCallback* structure describing the procedure which V.22bis will call as 16-bit linear PCM samples are generated by the V.22bis Transmit algorithm. The callback procedure has the following declaration:

```
void    (*pCallback) ( void * pCallbackArg,
                       v22bis_eStatus Status,
                       UWord16 * pSamples,
                       UWord16 NumberSamples);
```

The callback procedure parameter, ***pCallbackArg***, is supplied by the user in the *v22bis_sTXCallback* structure; this value is passed back to the user during the call to the callback procedure. Typically, *pCallbackArg* points to context information used by the user's callback procedure.

Status Takes one of the values from the enumeration *v22bis_eStatus*. The enumeration is listed below for reference.

V22BIS_1200BPS_CONNECTION_ESTABLISHED,

V22BIS_1200BPS_CONNECTION_ESTABLISHED,

V22BIS_2400BPS_CONNECTION_ESTABLISHED,

V22BIS_CONNECTION_LOST,

V22BIS_DATA_AVAILABLE,

V22BIS_RETRAINING,

V22BIS_CARRIER_LOST_IN_DATAMODE

pSamples A pointer to the buffer containing linear 16-bit PCM samples generated by V.22bis transmitter.

NumberSamples Represents the number of samples in the buffer pointed to by ***pSamples***

pRXCallback A pointer to a *v22bis_sRXCallback* structure describing the procedure which *v22bis* will call as data bits are received by the V.22bis Receive algorithm. The callback procedure has the following declaration:

```
void    (*pCallback) ( void * pCallbackArg,
                       v22bis_eStatus   Status,
                       UWord16 * pBits,
                       UWord16 NumberBits);
```

The callback procedure parameter, ***pCallbackArg***, is supplied by the user in the *v22bis_sRXCallback* structure; this value is passed back to the user during the call to the callback procedure. Typically, *pCallbackArg* points to context information used by the user's callback procedure.

Status is similar to that described as part of *pTXCallback*

pBits is a pointer to the buffer containing received bits

NumberBits represents the number of received bits pointed to by *pBits*

Only the Least Significant Bits are valid.

Example Transmit and Receive callback procedures are shown in **Code Example 3-2** for reference; you must write your own callback procedure.

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Transmit Callback Procedure This callback routine sets a global flag, *connection_established*, to indicate to the main routine that the connection is established with the remote modem. If the data is available, the generated samples are written to the codec for transmission at 7200 samples/second.

Code Example 3-2. Sample Transmit Callback Procedure

```
void TXCallbackRoutine ( void * pCallbackArg, v22bis_eStatus Status,
                         Word16 * pSamples, UWord16 NumberSamples)
{
      Word16  i;
      if ( (Status == V22BIS_1200BPS_CONNECTION_ESTABLISHED) ||
            (Status == V22BIS_2400BPS_CONNECTION_ESTABLISHED) )
      {
            connection_established = 1;
      }


      if ( Status == V22BIS_DATA_AVAILABLE)
      {
      for  ( i = 0; i < NumberSamples; i++)
      CodecTxBuffer[i] = pSamples[i];
            /* Write the samples to the codec */
            write(CodecTx, CodecTxBuffer, NumberSamples);
          }
      }
```

Receive Callback Procedure This callback routine sets a global flag, *connection_lost*, to indicate to the main routine that the connection is lost with the remote modem. If there are valid received bits, they are written to the output buffer. In **Code Example 3-3**, *Outbuffer* refers to the output buffer of type ***char*** and *num_rx_bytes* is used to count the number of bytes received.This callback routine also updates a global variable, *carrierlosttime*, used to keep track of carrier lost time. If carrier is lost continuously for more than 5 seconds, then the global flag *conection_lost* will be set to indicate that the connection might have been lost.

Code Example 3-3. Sample Receive Callback Procedure

```
      void RXCallbackRoutine (void * pCallbackArg, v22bis_eStatus Status,
                              char * pBits, UWord16 NumberBits)
      {
          UWord16  num_bytes, i;

          if ( Status == V22BIS_CONNECTION_LOST)
```


For More Information On This Product,
Go to: www.freescale.com

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

```
                connection_lost = 1;


            if ( Status == V22BIS_DATA_AVAILABLE)
            {
                 carrierlosttime = 0;
                 num_bytes =  NumberBits >> 3;
                 for ( i = 0; i < num_bytes; i++);
                 {
                     Outbuffer[num_rx_bytes] = *pBits++;
                     num_rx_bytes++;
                 }
            }

           if ( Status == V22BIS_CARRIER_LOST_IN_DATAMODE)
           {
                carrierlosttime++;

                /* If carrier drop is for more than 5 sec terminate the
                   modem connection. 5 sec is chosen just to validate the
                   API for carrier drop. User needs to chose a specific time
                   for his/her own application. The Rx Callback is made every
                   12 samples and the sampling rate is 7200 Hz. For five
                   seconds the time count is 7200*5/12 = 3000 */

                if (carrierlosttime >= 3000)
                    connection_lost = 1;
           }
      }
```

For details on the structures above, please refer to the C header file *v22bis.h*, shown in **Code Example 3-1** as a reference.

Returns: Upon successful completion, a value of “TRUE” will be returned. Otherwise, a value of “FALSE” will be returned.

Special Considerations: This function must be called first and called only once by the user to initialize the V.22bis modem algorithm. The current implementation of V.22bis requires memory sections to be allocated via linker commands. For details on linking applications, see **Chapter 5**.

Code Example 3-4. Use of the *v22bisInit* Interface

```
#include “v22bis.h”
#include “mem.h”

extern  void TXCallbackRoutine (void * pCallbackArg, v22bis_eStatus Status,
                                Word16 * pSamples, UWord16 NumberSamples);
extern  void RXCallbackRoutine (void * pCallbackArg, v22bis_eStatus Status,
                                char * pBits, UWord16 NumberBits);
v22bis_sTXCallback  v22bis_TXCallback; /* Transmit callback routine */
v22bis_sRXCallback  v22bis_RXCallback; /* Receive callback routine */
```


For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

```
void test_v22bis (void)
{
     v22bis_sConfigure    *pConfig;
     v22bis_sHandle       *pV22bis;
     Result               result;
     UWord16              modem_config;
     v22bis_sHandle       pV22bis;

     /* Modem configuration parameters */
     modem_config = 0;
     modem_config = ( V22BIS_ANSWER_MODEM | V22BIS_GUARD_TONE_DISABLE |
                      V22BIS_SELF_RETRAIN_ENABLE |
                        V22BIS_V14_ENABLE_ASYNC_MODE);
     /* Allocate memory for the init structure */
      pConfig = memMallocEM( sizeof(v22bis_sConfigure));
       v22bis_TXCallback.pCallback = TXCallbackRoutine;
      v22bis_RXCallback.pCallback = RXCallbackRoutine;

      /* Initialize the init structure */
      pConfig ->Flags = modem_config;
      pConfig ->TXCallback = v22bis_TXCallback;
      pConfig ->RXCallback = v22bis_RXCallback;
       /* Call Create routine */
       pV22bis = v22bisCreate ( v22bis_sConfigure * pConfig)
      /* Call Init routine */
      result = v22bisInit (pV22bis, pConfig);
}
```

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

For More Information On This Product,
Go to: www.freescale.com

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

3.3.3 *v22bisTXDataInit*

Call(s):

```
Result v22bisTXDataInit   (v22bis_sHandle *pV22bis, char *pBits,
                                UWord16 NumBytes);
```

Required Header: *v22bis.h*

Arguments:

Table 3-3. *v22bisTXDataInit* Arguments

pV22bis	in	Pointer to an instance of V.22 bis.
pBits	in	Pointer to the user data bits to be transmitted by the V.22bis
NumBytes	in	The number of data bytes to be transmitted

Description: This module should be called to initialize the modem transmitter with the data to be transmitted to the remote modem. Transmission of actual data (user bits) will begin only after the modem establishes the connection with the remote modem. While the modem is transmitting data, the buffer pointed by **pBits* should not be disturbed. When the transmission of current data is complete, the user can call this module with the next data available. See *v22bisTX* (**Section 3.3.4**) for details on the use of *v22bisTXDataInit*. For complete details on the use of *v22bisTXDataInit*, please refer to the files **...nos\applications\v22interopcans.c** or **...nos\applications\v22\interopcall.c**.

Returns: Upon successful completion, *v22bisTXDataInit* will return “PASS”; if any error occurred, it will return “FAIL”.

Code Example 3-5. Use of *v22bisTXDataInit* Interface

```
#include “v22bis.h”
#include “mem.h”

extern  void TXCallbackRoutine (void * pCallbackArg, v22bis_eStatus Status,
                                Word16 * pSamples, UWord16 NumberSamples);
extern  void RXCallbackRoutine (void * pCallbackArg, v22bis_eStatus Status,
                                char * pBits, UWord16 NumberBits);

v22bis_sTXCallback  v22bis_TXCallback; /* Transmit callback routine */
v22bis_sRXCallback  v22bis_RXCallback; /* Receive callback routine */

void test_v22bis (void)
{
     v22bis_sConfigure    *pConfig;
     v22bis_sHandle       *pV22bis;
     Result               result;
     UWord16              modem_config;
     char                 bit_array[20] = “V22bis testing”;
     UWord16              NumberBytes = 14;   /* Number of bytes in the string */

       /* Modem configuration parameters */
```


For More Information On This Product,
Go to: www.freescale.com
Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

```
        modem_config = 0;
        modem_config = ( V22BIS_ANSWER_MODEM | V22BIS_GUARD_TONE_DISABLE |
                                V22BIS_SELF_RETRAIN_ENABLE |
        V22BIS_V14_ENABLE_ASYNC_MODE);
        /* Allocate memory for the init structure */
            pConfig = memMallocEM( sizeof(v22bis_sConfigure));
        v22bis_TXCallback.pCallback = TXCallbackRoutine;
        v22bis_RXCallback.pCallback = RXCallbackRoutine;


            /* Initialize the init structure */
            pConfig ->Flags = modem_config;
            pConfig ->TXCallback = v22bis_TXCallback;
            pConfig ->RXCallback = v22bis_RXCallback;

           /* Call Create routine */

           pV22bis = v22bisCreate ( v22bis_sConfigure * pConfig)

            /* Call Init routine */
            result = v22bisInit (pV22bis, pConfig);
              /* Keep calling v22bisRX till the status on the trasmit callback routine
        becomes Connection Established. Once the connection is established call
        v22bisTXDataInit as and when user data is available */

        result = v22bisTXDataInit ( pV22bis, bit_array, NumberBytes);
}
```

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

For More Information On This Product,
Go to: www.freescale.com

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

3.3.4 *v22bisTX*

Call(s):

```
Result v22bisTX (v22bis_sHandle *pV22bis)
```

Required Header: "v22bis.h"

Arguments:

Table 3-4. *v22bisTX* Arguments

pV22bis	in	Pointer to an instance of V.22 bis

Description: The V.22bis transmitter should be called to generate the modulated samples corresponding to the input (user) bits. Transmission of actual data (user bits) will begin only after the modem establishes the connection with the remote modem. Until the connection is established, the user application should keep calling only *v22bisRX* (the V.22bis **receiver**), as explained in **Section 3.3.5**.

When the transmitter has 12 samples, it will make a callback to the user application as described in *v22bisInit*, **Section 3.3.2**. These samples are to be transmitted to the codec at 7200 samples/second by the user application. During handshake mode, *v22bisRX* will make a callback to the user application with the return value "V22BIS_1200BPS_CONNECTION_ESTABLISHED" or "V22BIS_2400BPS_CONNECTION_ESTABLISHED". Once the connection is established, the user calls *v22bisTXDataInit* to initialize the transmitter with the data to be transmitted. As explained in **Code Example 3-6**, *v22bisTX* can be called by the user application and makes a callback when it has generated 12 samples. It generates 12 samples for every TWO input bits (in V.22, 1200bps) or FOUR input bits (in V.22, 2400bps). For complete details on the use of *v22bisTX*, please refer to the files **...nos\applications\v22\interopans.c** or **...nos\applications\v22\interopcall.c**.

Returns: While samples are being generated and transmitted for the valid user data, "PASS" will be returned. When all samples have been generated and transmitted, "FAIL" will be returned.

Special Considerations:

- In the handshake mode, the V.22bis receiver calls only the transmitter, so the application program need not call the *v22bisTX* routine.
- The *v22bisTX* routine should be called by the application program only in the data mode and when the modem is not retraining in the data mode.
- In the data mode, after handshake is complete, *v22bisTX* should be called only after calling *v22bisTXDataInit* to initialize the transmitter with the user data.
- Once the transmitter has been initialized with the user data by calling *v22bisTXDataInit*, keep calling *v22bisTX* until the return status is "FAIL". At this point, all user data has been transmitted. Calling *v22bisTXDataInit* again supplies the next data to the transmitter.
- In the asynchronous mode of operation, *v22bisTX* appends one start and one stop bit to every byte of user data. In the synchronous mode, no start and stop bits will be added.
- In the asynchronous mode of operation, if *v22bisTX* is called without any user data, stop bits will be transmitted every call. Even if the user data is not available, the application program should keep calling the *v22bisTX* routine to maintain the carrier synchronization with the remote modem.
- The *v22bisTX* routine should not be called when the modem is in retraining mode. During this mode, the receiver itself calls the transmitter and makes the transmit callback whenever 12 samples are generated.

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Code Example 3-6. Use of *v22bisTX* Interface

```
#include “v22bis.h”
#include “mem.h”

extern  void TXCallbackRoutine (void * pCallbackArg, v22bis_eStatus Status,
                                Word16 * pSamples, UWord16 NumberSamples);
extern  void RXCallbackRoutine (void * pCallbackArg, v22bis_eStatus Status,
                                char * pBits, UWord16 NumberBits);

v22bis_sTXCallback  v22bis_TXCallback; /* Transmit callback routine */
v22bis_sRXCallback  v22bis_RXCallback; /* Receive callback routine */

void test_v22bis (void)
{
     v22bis_sConfigure    *pConfig;
     v22bis_sHandle       *pV22bis;
     Result               result;
     UWord16              modem_config;
     char                 bit_array[20] = “V22bis testing”;
     UWord16              NumberBytes = 14;   /* Number of bytes in the string */

       /* Modem configuration parameters */
       modem_config = 0;
       modem_config = ( V22BIS_ANSWER_MODEM | V22BIS_GUARD_TONE_DISABLE |
                        V22BIS_SELF_RETRAIN_ENABLE | V22BIS_V14_ENABLE_ASYNC_MODE);
       /* Allocate memory for the init structure */
       pConfig = memMallocEM( sizeof(v22bis_sConfigure));
             v22bis_TXCallback.pCallback = TXCallbackRoutine;
             v22bis_RXCallback.pCallback = RXCallbackRoutine;


           /* Initialize the init structure */
           pConfig ->Flags = modem_config;
           pConfig ->TXCallback = v22bis_TXCallback;
           pConfig ->RXCallback = v22bis_RXCallback;

          /* Call Create routine */

          pV22bis = v22bisCreate ( v22bis_sConfigure * pConfig)

           /* Call Init routine */
           result = v22bisInit (pV22bis, pConfig);

       /* Keep calling v22bisRX till the status on the trasmit callback routine
       becomes Connection Established. Once the connection is established call
       v22bisTXDataInit as and when user data is available */
       result = v22bisTXDataInit ( pV22bis, bit_array, NumberBytes);

       /* Keep calling till result becomes FAIL */
             result = v22bisTX (pV22bis);

}
```

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

For More Information On This Product,
Go to: www.freescale.com

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

3.3.5 *v22bisRX*

Call(s):

```
Result v22bisRX ( v22bis_sHandle *pV22bis, Word16* pSamples,
                  UWord16 NumberSamples);
```

Required Header: *v22bis.h*

Arguments:

Table 3-5. *v22bisRX* Arguments

pV22bis	in	Pointer to an instance of V.22 bis
pSamples	in	Pointer to the 16-bit linear digital PCM samples from the analog signal
NumberSamples	in	The number of data samples pointed to by *pSamples*

Description: The user application should call *v22bisRX* with samples collected from the codec. The sampling rate should be 7200Hz. Until the connection is established, the user application should keep calling the *v22bisRX* function only.

During the handshake mode (before the connection is established), the V.22bis receiver itself calls the V.22bis transmitter to generate the samples needed for handshake. Once the connection is established, the *v22bisRX* will make a callback to the user application with the return value "V22BIS_1200BPS_CONNECTION_ESTABLISHED" or "V22BIS_2400BPS_CONNECTION_ESTABLISHED". If there is any error during the handshake, the callback procedure will have a return value "V22BIS_CONNECTION_LOST".

During retraining in data mode, the return value on the callback is "V22BIS_RETRAINING". In retraining mode, the receiver itself calls the transmitter to generate the handshake signals.

If the carrier is lost while the modem is in the data mode, "V22BIS_CARRIER_LOST_IN_DATAMODE" will be returned on the callback. The user application can continue to count the continuous carrier drops and can timeout accordingly.

The *v22bisRX* demodulates 12 samples at a time, decodes two or four bits and calls back the user application when eight bits are collected. The *v22bisRX* function can be called whenever the user receives samples to process. For complete details on the use of *v22bisRX*, please refer to the files: **...nos\application\v22\interopans.c** or **...nos\applications\v22\interopcall.c**.

Returns: Upon successful completion, *v22bisRX* will return "PASS"; if any error occurred, *v22bisRX* will return "FAIL".

Code Example 3-7. Use of *v22bisRX* Interface

```
#include "v22bis.h"
#include "mem.h"

extern  void TXCallbackRoutine (void * pCallbackArg, v22bis_eStatus Status,
                                Word16 * pSamples, UWord16 NumberSamples);
extern  void RXCallbackRoutine (void * pCallbackArg, v22bis_eStatus Status,
                                char * pBits, UWord16 NumberBits);
```

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

```
v22bis_sTXCallback  v22bis_TXCallback; /* Transmit callback routine */
v22bis_sRXCallback  v22bis_RXCallback; /* Receive callback routine */

void test_v22bis (void)
{
      v22bis_sConfigure         *pConfig;
      v22bis_sHandle            *pV22bis;
      Result                    result;
      UWord16                   modem_config;
      Word16                    sample_array[12];
      UWord16                   NumberSamples = 12; /* 12 samples in the buffer */

      /* Modem configuration parameters */
            modem_config = 0;
            modem_config = ( V22BIS_ANSWER_MODEM | V22BIS_GUARD_TONE_DISABLE |
                             V22BIS_SELF_RETRAIN_ENABLE |
      V22BIS_V14_ENABLE_ASYNC_MODE);
      /* Allocate memory for the init structure */
      pConfig = memMallocEM( sizeof(v22bis_sConfigure));

            v22bis_TXCallback.pCallback = TXCallbackRoutine;
            v22bis_RXCallback.pCallback = RXCallbackRoutine;


          /* Initialize the init structure */
          pConfig ->Flags = modem_config;
          pConfig ->TXCallback = v22bis_TXCallback;
          pConfig ->RXCallback = v22bis_RXCallback;

         /* Call Create routine */

         pV22bis = v22bisCreate ( v22bis_sConfigure * pConfig)

          /* Call Init routine */
          result = v22bisInit (pV22bis, pConfig);

      result = v22bisRX ( pV22bis, sample_array, UWord16 NumberSamples);

}
```

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

For More Information On This Product,
Go to: www.freescale.com

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

3.3.6 *v22bisControl*

Call(s):

```
Result v22bisControl (UWord16 Command);
```

Required Header: *v22bis.h*

Arguments:

Table 3-6. *v22bisControl* Arguments

Command	in	The command to be executed by the *v22bisControl* procedure

Description: The *v22bisControl* function provides control functions to the V.22bis data pump algorithm.

The parameter *pV22bis* must have been generated from a call to *v22bisCreate*. Currently, no control commands are supported for V.22bis; this is for future use.

Returns: If used, this function always returns “PASS”.

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

3.3.7 *v22bisDestroy*

Call(s):

```
void v22bisDestroy (v22bis_sHandle *pV22bis);
```

Required Header: *v22bis.h*

Arguments:

Table 3-7. *v22bisDestroy* Arguments

pV22bis	in	Handle to an instance of V.22bis generated by a call to *v22bisCreate*

Description: The *v22bisDestroy* function destroys the instance of the V.22bis data pump originally created by a call to *v22bisCreate*.

Returns: This function always returns “PASS”.

Code Example 3-8. Use of *v22bisDestroy* Interface

```
#include “v22bis.h”
{
        ...
        v22bisDestroy ( pV22bis);


}
```

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

For More Information On This Product,
Go to: www.freescale.com

Chapter 4
Building the V.22bis Library

4.1 Building the V.22bis Library

The V.22bis library combines all of the components described in previous sections into one library: *v22bis.lib*. To build this library, a Metrowerks' CodeWarrior project, *v22bis.mcp*, is provided. This project and all the necessary components to build the V.22bis library are located in the ...*modem\v22* directory of the SDK directory structure.

There are two methods to execute system library project build: dependency build and direct build.

4.1.1 Dependency Build

Dependency build is the easiest approach and requires no additional work on the user's part. If you add the V.22bis library project, *v22bis.mcp*, to your application project as shown in **Figure 4-1**, the V.22bis library will automatically build when the application is built.

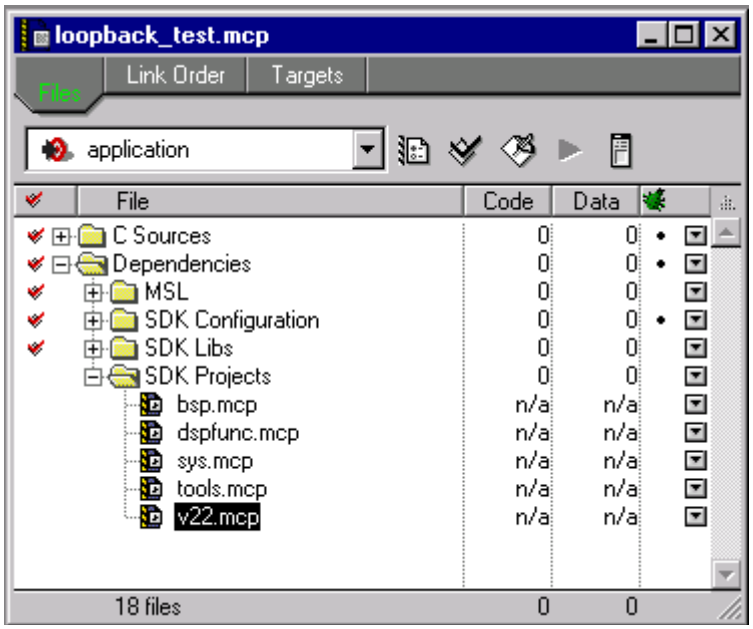

Figure 4-1. Dependency Build for V.22bis Library

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

4.1.2 Direct Build

Direct build allows you to build a V.22bis library independently of any other build. Follow these steps:

Step 1. Open *v22bis.mcp* project, as shown in **Figure 4-2**.

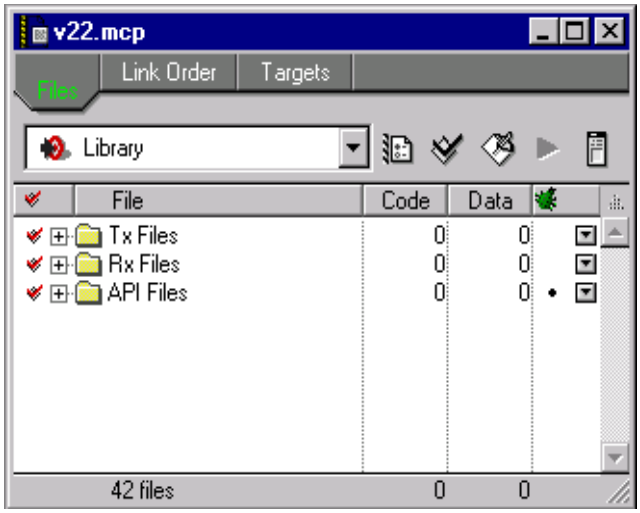

Figure 4-2. *v22bis.mcp* Project

Step 2. Execute the build by pressing function key [F7] or by choosing the *Make* command from the Project menu; see **Figure 4-3**.

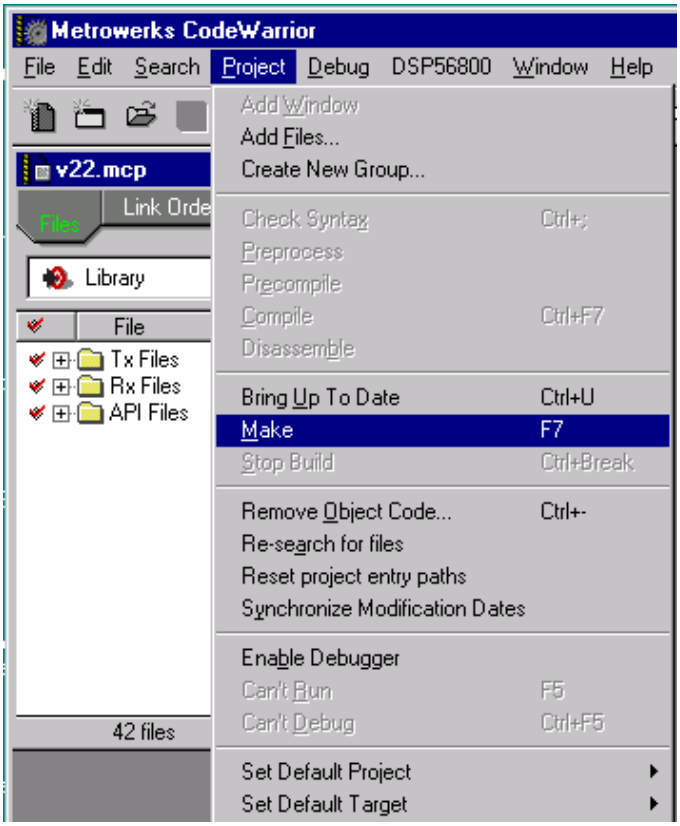

Figure 4-3. Execute Make

If the build is successful, a *v22bis.lib* library file is created in the **...\modem\v22\Debug** directory.

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Chapter 5
Linking Applications with the V.22bis Library

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

5.1 V.22bis Library

The V.22bis library includes a transmitter and a receiver. The library also includes APIs, which provide the interface between the user application and the V.22bis modules. To use the V.22bis library, APIs must be called in this order:

— v22bisInit(.....);
— v22bisRX (.....); /* Keep calling this module till it returns the message "*V22BIS_CONNECTION_ESTABLISHED*" . After the connection is established, call this module whenever there are samples to be processed */
— v22bisTxDataInit(.....); /* This module should be called by the user once the modem enters into the data mode */
— v22bisTX (.....); /* The user needs to call this module whenever there are bits to be transmitted */

5.1.1 Library Sections

The V.22bis Library contains program memory, internal data memory and data (internal or external memory) sections.

Internal data memory sections for V.22bis include:

- ***V22B_PROM*** - contains all tables for filter coefficients and the necessary data to be placed in the internal memory. This section also contains modulo buffers, so section alignment is needed. As a rule, if a section contains circular (modulo) buffers, that section should be aligned to the maximum circular buffer length. The largest modulo buffer size is 256 words, so the section should be aligned for 256 words; see **Code Example 5-1** for information on aligning the sections.
- **ROM_XMEM** - static data needed for V.22bis transmitter and receiver, which should be placed in the internal memory
- **RX_MEM** - temporary data variables needed by the V.22bis receiver, which should be placed in the internal memory and should be aligned to length 64 words.

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Data (internal or external) memory sections for V.22bis include:

- ***API*** - contains data needed for API routines
- ***TX_MEM*** - temporary variables needed by the V.22bis transmit routines

Note: All of the memory sections are grouped together for placement in different memory regions on the target. The memory regions and the grouping order of the sections under each region are shown in **Code Example 5-1** for reference. The order of the sections within each memory region should not be altered, and, if altered, the V.22bis library may not function as expected. For details, see the *linker.cmd* file, found in the **modem\v22\test\Config** directory in the SDK.

Code Example 5-1. V.22bis Library Sections

```
# Linker.cmd file for DSP56824EVM External RAM
# using both internal and external data memory (EX = 0)
# and using external program memory (Mode = 3)

MEMORY {
       .pram   (RWX) : ORIGIN = 0x0000, LENGTH = 0xFF80  # ? external program memory
       .avail  (RW)  : ORIGIN = 0x0000, LENGTH = 0x0030  # available
       .cwregs (RW)  : ORIGIN = 0x0030, LENGTH = 0x0010  # C temp registrs in
                                 CodeWarrior
       .im1    (RW)  : ORIGIN = 0x0040, LENGTH = 0x07C0  # data 1
       .rom    (R)   : ORIGIN = 0x0800, LENGTH = 0x0800  # internal data ROM
       .im2    (RW)  : ORIGIN = 0x1000, LENGTH = 0x0600  # data 2
       .hole   (R)   : ORIGIN = 0x1600, LENGTH = 0x0A00  # hole
       .data   (RW)  : ORIGIN = 0x2000, LENGTH = 0xC000  # data segment
       .em     (RW)  : ORIGIN = 0xE000, LENGTH = 0x1000  # data 3
       .stack  (RW)  : ORIGIN = 0xF000, LENGTH = 0x0F80  # stack
       .onchip1(RW)  : ORIGIN = 0xFF80, LENGTH = 0x0040  # on-chip peripheral
                                 registers
       .onchip2(RW)  : ORIGIN = 0xFFC0, LENGTH = 0x0040  # on-chip peripheral
                                 registers
}

FORCE_ACTIVE {FconfigInterruptVector}

SECTIONS {
       #
       # Data (X) Memory Layout
       #
              _EX_BIT     = 0;

       # Internal Memory Partitions (for mem.h partitions)
       _NUM_IM_PARTITIONS = 1;  # .im1 and .im2

       # External Memory Partition (for mem.h partitions)
       _NUM_EM_PARTITIONS = 1;   # .em

       .main_application_code :
       {
```


For More Information On This Product,
Go to: www.freescale.com

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

```
# .text sections
#  config.c MUST be placed first, otherwise the Interrupt Vector
#  configInterruptVector will not be located at the correct address, P:0x0000

config.c (.text)
* (.text)
* (rtlib.text)
* (fp_engine.text)
* (user.text)

} > .pram

.main_application_data :
{
#
# Define variables for C initialization code
#
F_Xdata_start_addr_in_ROM = ADDR(.rom) + SIZEOF(.rom) / 2;
F_StackAddr               = ADDR(.stack);
F_StackEndAddr            = ADDR(.stack) + SIZEOF(.stack) / 2  - 1;

F_Xdata_start_addr_in_RAM = .;
#
# Memory layout data for SDK INCLUDE_MEMORY (mem.h) support
#
FmemEXbit = .;
      WRITEH(_EX_BIT);
FmemNumIMpartitions = .;
      WRITEH(_NUM_IM_PARTITIONS);
FmemNumEMpartitions = .;
      WRITEH(_NUM_EM_PARTITIONS);
FmemIMpartitionList = .;
#     WRITEH(ADDR(.im1));
#     WRITEH(SIZEOF(.im1) / 2);
      WRITEH(ADDR(.im2));
      WRITEH(SIZEOF(.im2) / 2);
FmemEMpartitionList = .;
      WRITEH(ADDR(.em));
      WRITEH(SIZEOF(.em) /2);

# .data sections

* (.data)
* (fp_state.data)
* (rtlib.data)


# V22bis external data starts here
#---------------------------------

* (TX_MEM.data)
* (API.data)

# V22bis external data ends here
#-------------------------------
```

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

For More Information On This Product,
Go to: www.freescale.com

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

```
        F_Xdata_ROMtoRAM_length = 0;

        F_bss_start_addr = .;
        _BSS_ADDR = .;

        * (rtlib.bss.lo)
        * (.bss)


        # V22bis bss starts here
        #-----------------------

        * (TX_MEM.bss)
        * (API.bss)

        # V22bis bss ends here
        #---------------------

        F_bss_length = . - _BSS_ADDR;  # Copy DATA


        } > .data

        FArchIO   = ADDR(.onchip2);


        # Internal memory allocation for V22bis
        #--------------------------------------

        .V22bis_internal_data :
        {

        .=ALIGN(0x100);         #For modulo buffer of size 256
        * (V22B_PROM.data)
        * (ROM_XMEM.data)

        .=ALIGN(0x40);          #For modulo buffer of size 64
        * (RX_MEM.data)

        # .bss sections
        * (V22B_PROM.bss)
        * (ROM_XMEM.bss)
        * (RX_MEM.bss)

        } > .im1

}
```

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Chapter 6
V.22bis Applications

6.1 V.22bis Test and Demo Applications

To verify the V.8 algorithm, test and demo applications have been developed. Refer to the **Targeting Motorola DSP568xx Platform** manual for the DSP you are using to see if the test and demo applicaitons are available for your target.

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Chapter 7
License

7.1 Limited Use License Agreement

LIMITED USE LICENSE AGREEMENT

PLEASE READ THIS AGREEMENT CAREFULLY BEFORE USING THIS SOFTWARE. BY USING OR COPYING THE SOFTWARE, YOU AGREE TO THE TERMS OF THIS AGREEMENT.

The software in either source code form ("Source") or object code form ("Object") (cumulatively hereinafter "Software") is provided under a license agreement ("Agreement") as described herein. Any use of the Software including copying, modifying, or installing the Software so that it is usable by or accessible by a central processing unit constitutes acceptance of the terms of the Agreement by the person or persons making such use or, if employed, the employer thereof ("Licensee") and if employed, the person making such use hereby warrants that they have the authority of their employer to enter this license agreement,. If Licensee does not agree with and accept the terms of this Agreement, Licensee must return or destroy any media containing the Software or materials related thereto, and destroy all copies of the Software.

The Software is licensed to Licensee by Motorola Incorporated ("Motorola") for use under the terms of this Agreement. Motorola retains ownership of the Software. Motorola grants only the rights specifically granted in this Agreement and grants no other rights. Title to the Software, all copies thereof and all rights therein, including all rights in any intellectual property including patents, copyrights, and trade secrets applicable thereto, shall remain vested in Motorola.

For the Source, Motorola grants Licensee a personal, non-exclusive, non-assignable, revocable, royalty-free right to use, copy, and make derivatives of the Source solely in a development system environment in order to produce object code solely for operating on a Motorola semiconductor device having a central processing unit ("Derivative Object").

For the Object and Derivative Object, Motorola grants Licensee a personal, non-exclusive, non-assignable, revocable, royalty-free right to copy, use, and distribute the Object and the Derivative Object solely for operating on a Motorola semiconductor device having a central processing unit.

Licensee agrees to: (a) not use, modify, or copy the Software except as expressly provided herein, (b) not distribute, disclose, transfer, sell, assign, rent, lease, or otherwise make available the Software, any derivatives thereof, or this license to a third party except as expressly provided herein, (c) not remove obliterate, or otherwise defeat any copyright, trademark, patent or proprietary notices, related to the

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Software (d) not in any form export, re-export, resell, ship or divert or cause to be exported, re-exported, resold, shipped, or diverted, directly or indirectly, the Software or a direct product thereof to any country which the United States government or any agency thereof at the time of export or re-export requires an export license or other government approval without first obtaining such license or approval.

THE SOFTWARE IS PROVIDED ON AN "AS IS" BASIS AND WITHOUT WARRANTY OF ANY KIND INCLUDING (WITHOUT LIMITATION) ANY WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL MOTOROLA BE LIABLE FOR ANY LIABILITY OR DAMAGES OF ANY KIND INCLUDING, WITHOUT LIMITATION, DIRECT OR INDIRECT OR INCIDENTAL OR CONSEQUENTIAL OR PUNITIVE DAMAGES OR LOST PROFITS OR LOSS OF USE ARISING FROM USE OF THE SOFTWARE OR THE PRODUCT REGARDLESS OF THE FORM OF ACTION OR THEORY OF LIABILITY (INCLUDING WITHOUT LIMITATION, ACTION IN CONTRACT, NEGLIGENCE, OR PRODUCT LIABILITY) EVEN IF MOTOROLA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. THIS DISCLAIMER OF WARRANTY EXTENDS TO LICENSEE OR USERS OF PRODUCTS AND IS IN LIEU OF ALL WARRANTIES WHETHER EXPRESS, IMPLIED, OR STATUTORY, INCLUDING IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR PARTICULAR PURPOSE.

Motorola does not represent or warrant that the Software is free of infringement of any third party patents, copyrights, trade secrets, or other intellectual property rights or that Motorola has the right to grant the licenses contained herein. Motorola does not represent or warrant that the Software is free of defect, or that it meets any particular requirements or need of the Licensee, or that it conforms to any documentation, or that it meets any standards.

Motorola shall not be responsible to maintain the Software, provide upgrades to the Software, or provide any field service of the Software. Motorola reserves the right to make changes to the Software without further notice to Licensee.

The Software is not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Software could create a situation where personal injury or death may occur. Should Licensee purchase or use the Software for any such unintended or unauthorized application, Licensee shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the Software.

The term of this Agreement is for as long as Licensee uses the Software for its intended purpose and is not in default of any provisions of this Agreement. Motorola may terminate this Agreement if Licensee is in default of any of the terms and conditions of this Agreement.

This Agreement shall be governed by and construed in accordance with the laws of the State of Arizona and can only be modified in a writing signed by both parties. Licensee agrees to jurisdiction and venue in the State of Arizona.

By using, modifying, installing, compiling, or copying the Software, Licensee acknowledges that this Agreement has been read and understood and agrees to be bound by its terms and conditions. Licensee agrees that this Agreement is the complete and exclusive statement of the agreement between Licensee and Motorola and supersedes any earlier proposal or prior arrangement, whether oral or written, and any other communications relative to the subject matter of this Agreement.

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Index

C

Calling modem 1-2

D

Dependency Build 4-1
Direct Build 4-2
DSP x
DSP56800 Family Manual xi
DSP56824 User's Manual xi

E

Embedded SDK Programmer's Guide xi

I

I/O x
I/O Services 3-1, 5-1
IDE x

L

Linking Applications 5-1
LSB x

M

MAC xi
MIPS xi
MSB xi

O

OMR xi
OnCE xi

P

PC xi

S

SDK xi
SP xi
SPI xi
SR xi
SRC xi

V

V.22bis
    Applications 6-1
V22bis
    Library 5-1
    Performance 1-2
V22bis algorithm 2-2
v22bisControl 3-17
    Arguments 3-17
v22bisCreate 3-5
    Arguments 3-5
v22bisDestroy 3-18
    Arguments 3-18
    Interface 3-18
v22bisInit 3-6
    Arguments 3-6
    Interface 3-9
v22bisRX 3-15
    Arguments 3-15
    Interface 3-15
v22bisTX 3-13
    Arguments 3-11, 3-13
    Interface 3-11, 3-14
v22bisTXDataInit 3-11

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC. 2005

Motorola reserves the right to make changes without further notice to any products herein. Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters which may be provided in Motorola data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including “Typicals” must be validated for each customer application by customer’s technical experts. Motorola does not convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part. Motorola and the Stylized M Logo are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

MOTOROLA and the Stylized M Logo are registered in the US Patent & Trademark Office. All other product or service names are the property of their respective owners. © Motorola, Inc. 2002.

How to reach us:
USA/EUROPE/Locations Not Listed: Motorola Literature Distribution; P.O. Box 5405, Denver, Colorado 80217. 1–303–675–2140 or 1–800–441–2447

JAPAN: Motorola Japan Ltd.; SPS, Technical Information Center, 3–20–1, Minami–Azabu. Minato–ku, Tokyo 106–8573 Japan. 81–3–3440–3569

ASIA/PACIFIC: Motorola Semiconductors H.K. Ltd.; Silicon Harbour Centre, 2 Dai King Street, Tai Po Industrial Estate, Tai Po, N.T., Hong Kong. 852–26668334

Technical Information Center: 1–800–521–6274

HOME PAGE: http://www.motorola.com/semiconductors/

MOTOROLA

SDK108/D

**For More Information On This Product,
Go to: www.freescale.com**